Andrew Sun

006159649

One notable obstacle that I had to overcome was figuring out how to keep track of the number of times I skipped a paragraph due to @P@. It was difficult for me to keep track of the times that the program had skipped a paragraph and how to stop the program from skipping another paragraph if another @P@ was passed through. I eventually figured out a solution by adding in a few booleans to keep track of both skipping the paragraph and if we need to ignore the other commands due to a line being skipped.

Pseudocode:

Int render

If lineLength < 1

Output 2 for error

Set up variables and vector

While there is still characters in inf

Unget first char from the loop

Reset skipLine

Get the first char from inf

Check if string is 3 char

Check if string is @P@ and we have not skipped paragraph yet

Set starting word to false (prevent space from being added at start)

Set skipLine true (prevent rest of program from running)

Set hasSkipped true (only skip one paragraph)

Skip paragraph

Reset variables

Reset length count

If string is @P@ and we have skipped

Ignore rest of the commands and clear the string

Check if the string is not the first word in the line and has punctuation preceding it

Output a space

Reset hasPunct variable

Update length count

Check if current string can be fit on existing line

Check if string is not the first word in the line

Output space before the string

Update length

If string is first word

Set firstWord variable to false

Output all char in the string

If the string has characters

Reset hasSkipped variable

Check if last char of string is a punctuation

Set hasPunct to true

Else

Set hasPunct to false

Update length count

Repeat of the code block above but output.size() <= remainingLength to fix a hyphen bug

Else if we need to skip line and input string into next line

Check if the string is not the first word in the input

Reset length count

Output \n

startingWord false

Output all char in string

Check last char of string for punctuation

Reset variables

Update length count

Else (case where string is larger than lineLength)

Set up counter to count number of extra strings

Check if string is not the first word in input

Set returnOne to true

Output \n

startingWord false

Output as much of the string as we can in first line

Output \n

Output rest of string and update count

Check if last char is punctuation

Update length count by subtracting count from lineLength

Output last \n

If returnOne is true

Return 1

Else

Return 0

vector<char> stringGetter

Temporarily set newLine to false

Set up vector

Set up char

While there is char available in inf

If char is not a space or \n

Add char to vector

If the char is ‘-’ (word subgroup)

hasHyphen true

Break

If char is space or \n

Break

If there is punctuation

Check if string can fit in currentLine-1 (for space)

If can’t, newLine true

If no punctuation

Check if string can fit in currentLine

If can’t, newLine true

If string is larger than lineLength

returnOne = true

Return result

Test cases: (input, lineLength)

1. Test test test test \n\n test, 4 //check program ability to ignore spaces
2. Testy, 4 //check output and also int return
3. Test. test test //check 2 spaces after period
4. Test? Test test //check 2 spaces after punctuation, also test for other punctuation cases (!,?,:)
5. Test, -1 //check error output
6. Test @P@ test, 4 //check new paragraph function
7. Test @P@ @P@ @P@ test, 4 //check if program ignores following @P@
8. test@P@ test, 20 //check if program ignores @P@ command because it is part of string
9. Test..test, 10 //check program ability to ignore punctuations
10. -----, 4 //check program ability to differentiate word subgroups
11. Test-test, 6 //check program ability to differentiate word subgroups